A CACHE SERVER NETWORK

TECHNICAL FIELD

The present invention relates to a method and a device for a cache server system. In particular the invention relates to a method for predicting information flow on the internet and other data networks comprising cache servers, and to a network implementing the method.

BACKGROUND OF THE INVENTION AND PRIOR ART

Data traffic in existing networks is constantly increasing. This is particularly the case for internet traffic, mainly due to the rapid increase in the number of users and the increase of information available on the internet. As a result of this increase in data traffic it is unavoidable that data traffic and server capacity sometimes hits bottlenecks.

A straightforward way of avoiding or reducing bottlenecks is to increase the capacity of servers and transmission lines. This, however, is very costly, since the capacity then must be adapted to an expected maximum throughput of data traffic.

Another way of lowering the requirement on servers and transmission lines is to cache information closer to the user. In other words caching, i.e. storing of replicated data in several locations and thereby closer to the users, increases the total network response capacity.

Thus, if data requested by a user can be found at a location which in some meaning is closer to the user than the location at which the original data is stored, time and capacity will be saved in the overall network. This will in turn result in lower costs for the overall system.

There are many different ways of configuring cache servers in a global data network, such as the Internet. For example, in a simple cache system, a cache server is connected to a user or to a group of users and data demanded by the user(s) is stored for

further demand for a fixed period of time, or until the cache memory is filled, when the data is replaced according to some suitable algorithm, such as first in first out (FIFO).

In another configuration, a meta server is connected to a number of such cache servers. The meta server then keeps track of what data is stored in the different cache servers. Thus, when a user demands some particular data, the cache server to which it is connected is first searched for the data.

If the data cannot be found in that cache server, it is checked if the data is available in any of the cache servers connected to the meta server. If that is the case, i.e. the data is available in any of the servers constituting the group of servers connected to the meta server, the data is fetched from that server, instead of from the location where the data originally is stored.

Thus, when data not can be found in any of the cache servers, it must be requested from the original source. This is of course not desired, and in particular not if the request is issued when the network over which data is to be retrieved has a high load.

SUMMARY

It is an object of the present invention to provide a method and a system by means of which data can be cached in a more efficient manner and by means of which hit rates can be increased.

This object and others are obtained by means of providing a forecast function, which can be implemented in a particular forecast caching server. The addition of such a function enables the cache system to cache data that have a higher probability of being demanded than is the case for conventional cache servers/ cache server systems.

Thus, the forecast function instructs cache servers to which it is connected to pre-fetch data or to store or not to store data that is fetched from original source servers to serve customers

in each area of caching servers served by the forecast caching server, via a certain protocol. The forecast caching server in a preferred embodiment keeps a database of all the addresses of all stored pages in all caching servers that it controls, as well as historic data.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will now be described in more detail by way of non-limiting examples and with reference to the accompanying drawings, in which:

- Fig. 1 is general view of a data system comprising caching servers.
- Fig. 2 is a view illustrating the configuration and functionality of a forecast caching server.

DESCRIPTION OF PREFERRED EMBODIMENTS

In Fig. 1 a data system is shown. The system comprises n1, n2, ... nk cache servers 101 with storage memory of m2,....mk megabyte, and network traffic throughput capacity of t1, t2, ... tk megabyte/second, and transaction capacities of tr1,tr2, ... trk transactions per second. A transaction being defined as certain instructions being processed with certain data. They all serve defined connections, e.g. transmission lines from companies 103 and from modem pools 105 for private customers.

The ultimate performance and lowest traffic costs in such a system are found when all data that are demanded more than once is cached in the cache servers. To get closer to this goal, forecasting is used. This is illustrated by the forecast cache control function located in a server 107.

The server 107, is thus connected to a number of caching servers 101 and has means for keeping a record of which information is stored in the different servers 101. For each caching server and cache address a probability function can be constructed, which in a preferred embodiment can be composed of the following factors of which the server 107 has knowledge:

* caching server identity
* address
* address level
* time
* date
* historic demand (when was the address asked for)
* historic update frequency (and update information from original sources)
* what other addresses where demanded a time period before and after the address was demanded (demand correlation)
* other correlated data (here data about e.g. football games can be stored such that during and after matches certain sports addresses/webpages are usually more demanded, and can be pre-fetched)

In Fig. 2 the functionality of the forecast server 107 is illustrated more in detail. Thus, since all data have names/addresses, e.g. for internet web pages http://www.cnn.com/, all addresses can be seen as branches on a tree, where the address gets longer as the tree is explored from the stem, to main branch to smaller branches etc. The branches, in difference from real trees, also have links to totally different addresses.

The control function in the server 107 can therefore have means for labelling all addresses with a level and give different levels different priorities. Below is an example of how the levels can be given.

http://www.cnn.com/ LEVEL 1,1
http://customnews.cnn.com/cnews/pna-auth.welcome LEVEL 2,3
http://www.cnn.com/WORLD/ LEVEL 2,2
http://www.cnn.com/WORLD/europe/ LEVEL 3,3
http://www.cnn.com/WORLD/europe/9803/21/AP000638.ap.html LEVEL 4,6
http://www.cnn.com/interactive - legal.html#AP LEVEL 5,2
http://www.lexis-nexis.com/lncc/about/terms.html LEVEL 6,4
http://www.cnn.com/US/
http://www.cnn.com/LOCAL/

http://www.cnn.com/WEATHER/
http://www.cnn.com/WEATHER/allergy/

LEVEL x,y means how many links that are passed from the starting page before the page (=x) is reached, and how many / there are (=y; excluding http://)

Based on this and the other parameters listed above a demand forecast is made using some suitable statistical method, and this is then compared to the existing stored data and the traffic, i.e. the existing demand capacity and the free capacity. The free traffic capacity is then used to pre-fetch pages/addresses based on the demand forecast.

Thus, for example, if there is a very high hit frequency on a particular address during the last 2 minutes and one or several addresses of that particular high frequency address has an (x)-value that is low, e.g. 1 or 2, it is likely that such an address will be demanded soon and the data on that address is pre-fetched and stored in one of the servers 101 and some data that has a lower probability of being demanded is discarded.

The forecast caching server 107 controls the caching servers (CS) 101 via a forecast control protocol, which can consist of at e.g. the following instructions.

* Address info (CS id, address)
* Store question (CS id, address)
* Store answer (CS id, address, yes/no)
* Fetch (CS id, address, levels) (i.e. the forecast cache server 107 can order a certain cache server 101 to fetch a main address and a number of levels down from the main address)
* Traffic question/answer (CS id, capacity, Mbyte traffic last period, period)
* Storage question/answer (CS id, capacity, Mbyte last period, period)

An important feature of the forecast caching system as described

herein is that the forecast caching server 107 always knows all addresses for which the web pages are stored in each caching server 101, the storage size and capacity, the traffic size and capacity.

Because it may be impossible or to expensive to store a demand function for all data that has been demanded, demand forecasts can be limited to addresses in number of x or y levels as described above. Because it may be impossible or to expensive to calculate a demand function for all data, decision rules for the forecast caching server for addresses without a demand function can be generalized from a limited set of data, such as:

- fill memory
- never store data the first time it is demanded unless there is free memory
- always store data the second time it is demanded
- first in first out if memory is filled

This function can also be delegated to and implemented in the caching servers 101.

Furthermore, in order to make it possible for the different cache servers 101 to inform each other about the currentness of web pages, an Update protocol can be defined, that can be used between any cooperating servers. This will result in that each source server, i.e. the server on which the original data is stored, has a log which states when every stored page and/or object of a page (e.g. picture, table, etc) was last updated.

Other severs can interrogate the source server (or his replicants), and compare to it's own log when the web page was stored to see if it has been updated. In this way, it is possible to make sure that a page is current without transferring all page data, only if any part of the page has been updated it has to be fetched again, or only the updated part/object. This saves capacity. Such exchange of information can be implemented with a protocol comprising the following instructions:

* Update question (Server id, page address)
* Update info (Server id, page address_1 , last updated, page object 1x, 1x last updated, page object 1y, 1y last updated, page address_2, last updated, page object 2x, 2x last updated, page object 2y, 2y last updated .....)

The update answer can of course be sent without a previous question, such as by agreement beforehand every minute/hour/etc.

In the network described above only one single forecast server controlling a plurality of cache servers is described. However, in a large network several forecast servers can be used. The forecast servers then controls a group of cache servers each and are interconnected with each other in a distributed manner.

In such a network solution a special protocol for exchange of information between the different forecast servers is used. Thus, by using such a protocol each forecast server has knowledge about or can ask for information on what data are stored in each cache server or cache server group.

The special protocol can also be used to send orders between the different forecast servers on which group of cache servers that is to store which data, or in another embodiment a negotiation is performed between the different forecast servers on which data that is to be stored in which cache server or group of cache servers.

In another preferred embodiment one in the multitude of forecast servers, which can be termed the main forecast server, is arranged to control the others, preferably by means of a special protocol similar to the one used for controlling the different cache servers. Such an arrangement will eliminate the need for a negotiation between the different forecast servers, since the main forecast server now will decide which data that will be stored at which location.

The use of a forecast caching function in a caching server system as described herein will thus lower traffic costs and

increases response times. This is due to the fact that the memory capacity of the caching servers in the system will be utilized more efficiently in terms of hit rate and that transmission capacity can be better utilized, since transmission capacity not currently used can be used for pre-fetching data having a high probability of soon being requested.